

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
Курский государственный университет

Кафедра Программного обеспечения  
и администрирования  
информационных систем

## Отчёт по эксплуатационной практике

Выполнил:

студент 213.1 группы

Козявин М.С.

Проверил:

ст. преподаватель кафедры ПОиАИС

Ураева Е.Е.

Курск 2023

## СОДЕРЖАНИЕ

Введение.....	3
1 Задание на период практики .....	4
1.1 Лабораторная работа №1. Основы языка C++ .....	4
1.2 Лабораторная работа №2. Система контроля версий Git .....	5
1.3 Лабораторная работа №3. Декомпозиция программы .....	5
1.4 Лабораторная работа №4. Библиотеки .....	7
2 Описание выполненной работы.....	8
2.1 Лабораторная работа №1. Основы языка C++ .....	8
2.2 Лабораторная работа №2. Система контроля версий Git .....	14
2.3 Лабораторная работа №3. Декомпозиция программы функциями..	22
2.4 Лабораторная работа №4. Библиотеки .....	34
3 Рабочий график проведения практики.....	43
Заключение .....	44
Список использованных источников .....	45

## ВВЕДЕНИЕ

Цель эксплуатационной практики в освоении приемов работы и программирования на языках C и C++, получении навыков использования системы контроля версий Git, работы со сторонними библиотеками, структурирования программы и проведения модульных тестов.

Данные навыки позволяют писать хорошо структурированный код, защищённый от ошибок за счёт покрытия модульными тестами, разрабатывать один проект в команде с другими разработчиками и иметь доступ к истории всех версий проекта.

Для достижения поставленной цели были поставлены следующие задачи:

1. Написать программу для построения гистограммы массива чисел. Научиться автоматически проверять программы по эталонному вводу и выводу.
2. Изучить приемы работы в системе контроля версий Git.
3. Научиться структурировать программу при помощи функций. Написать программу для вывода гистограммы как изображения в формате SVG. Научиться писать модульные тесты.
4. Написать программу построения гистограммы по данным из файла в сети.

## 1 Задание на период практики

### 1.1 Лабораторная работа №1. Основы языка C++

Задание:

1. Написать программу для построения гистограммы массива чисел.
2. Доработать программу в соответствии с вариантом.

Требования к выводу:

1. Подписи к столбцам выровнены до трех знакомест (можно считать, что в корзину больше 999 чисел не попадет).
2. Ширина всей гистограммы (подписи и звездочек в каждом столбце) должна укладываться в 80 символов. Если в корзину попало больше чисел, все столбцы нужно пропорционально сжать, чтобы выполнить условие.

Гистограмма — это наглядное графическое представление того, какие значения встречаются чаще или реже в исходных данных (как они распределены). Диапазон исходных данных делится на равные интервалы, для каждого интервала строится столбец. Высоты столбцов пропорциональны количеству значений, попавших в интервал. Таким образом сразу видно, какие значения встречаются чаще в целом (с точностью до ширины интервала) и насколько чаще по сравнению с другими (легко сравнить высоты визуально).

Гистограмма строится так: диапазон значений на входе делится на несколько равных интервалов (корзин, bins), подсчитывается количество чисел, попавших в каждый интервал, и для каждой корзины рисуется столбец, размер которого пропорционален количеству попавших в корзину чисел.

Например, на вход поступают оценки 10 студентов:

4 4 3 5 3 4 5 5 4 4

Пусть требуется построить гистограмму на три столбца. Диапазон значений на входе — от 3 до 5. Каждый из трех интервалов будет шириной  $(5 - 3) / 3 = 0,67$ , то есть интервалы будут  $[3; 3.67]$ ,  $[3.67; 4.34]$ ,  $[4.34; 5]$ . В первую корзину попадут тройки (2 шт.), во вторую — четверки (5 шт.), в третью — пятерки (3 шт.). Результат:

2 | \*\*

5 | \*\*\*\*\*

3 | \*\*\*

### Вариант 17

После ввода количества чисел предлагайте пользователю генерировать их. При положительном ответе заполните исходный массив при помощи функции `rand()`: каждый элемент должен быть суммой 12 ее результатов.

*Указание.* В начале программы добавьте `srand(time(0))`, чтобы случайные числа отличались между запусками программы (аналог `Randomize()` в Pascal). Для составления эталонного вывода замените `time(0)` на 42.

## 1.2 Лабораторная работа №2. Система контроля версий Git

Задание:

1. Вход в терминал и создание структуры каталогов.
2. Инициализация репозитория и настройка Git.
3. Создание коммитов
4. Игнорирование файлов
5. Просмотр истории
6. Откат изменений
7. Обмен кодом через удаленное хранилище
8. Совместная работа над проектом без конфликтов правок
9. Использование веток

## 1.3 Лабораторная работа №3. Декомпозиция программы

Задание:

Часть 1. Декомпозиция программы функциями

Программа для построения гистограммы из ЛР № 1 состоит из одной функции `main()` на более чем 100 строк, из-за чего в ней неудобно ориентироваться. Необходимо выделить части программы в функции:

Ввод чисел:

1. Принимает количество чисел, которое необходимо ввести;
2. Возвращает вектор чисел.

Поиск наибольшего и наименьшего значения:

1. Принимает вектор чисел;
2. Возвращает два результата — `min` и `max`.

Расчет гистограммы:

1. Принимает вектор чисел и количество корзин;
2. Возвращает вектор количеств чисел в каждой корзине;
3. Вызывает в процессе работы функцию поиска `min` и `max`.

Часть 2. Вывод гистограммы как изображения в формате SVG

Требуется вместо текстовой гистограммы рисовать картинку (рисунок 1).



Рисунок 1 – Пример вывода данных в формате SVG

Часть 3. Модульное тестирование

Модульный тест — это отдельная программа, которая изолированно проверяет части кода основной программы. Если желательно протестировать части сложного алгоритма, эти части должны быть оформлены в виде отдельных функций (говорят: код должен быть *тестируемым*).

Написать модульный тест для функции поиска минимума и максимума.

Вариант 17

Задавать автоматически прозрачность заливки каждого столбца гистограммы в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка. Сделать это можно, передавая процент прозрачности в

параметр `fill-opacity` в формате "0.7". 1 соответствует отсутствию прозрачности, 0 соответствует полной прозрачности (отсутствию цвета)

Для расчета прозрачности каждого  $i$ -го столбца `bins[i]` использовать формулу  $(bins[i]) / max\_count$  (рисунок 2).

```
1| █ — прозрачность 0.2
5| █████ — прозрачность 1.0
3| ███ — прозрачность 0.6
```

Рисунок 2 – Пример вывода данных в консоль с параметром прозрачности

## Лабораторная работа №4. Библиотеки

Задание:

Добавить возможность построения гистограммы по данным из файла из сети. Адрес файла задается аргументом командной строки программы. Если адрес не задан, читать данные со стандартного ввода, как раньше.

Пример строки запуска:

```
lab03.exe
http://uii.mpei.ru/study/courses/cs/lab03/marks.txt >marks.svg
```

### Вариант 17

Устанавливайте опцию `CURLOPT_FAILONERROR` и анализируйте результат `curl_easy_perform()`, чтобы проверить результат загрузки файла, а не выполнения запроса. Опишите в отчете способ тестирования.

## 2 Описание выполненной работы

### 2.1 Лабораторная работа №1. Основы языка C++

#### Текст программы

```
#include <iostream>

#include <vector>

using namespace std;

const char GIST_CHAR = '*';

const size_t SCREEN_WIDTH = 80;

const size_t MAX_GIST_WIDTH = SCREEN_WIDTH - 3 - 1;

const auto RAND_SEED = 42;

vector<double> getNumbersArray(size_t numbersCount);

vector<double> generateNumbersArray(size_t numbersCount);

vector<size_t> getBinsDistribution(vector<double> numbers,
size_t binCount);

void findMinMax(vector<double> numbers, double& min, double&
max);

void cmdOutput(vector<size_t> bins, size_t maxBinCount);

string formatLabel(size_t item);

string formatGist(size_t item, size_t maxBinCount);

size_t findMaxBinCount(vector<size_t> bins);

double getRandomNumber();

int main() {
    size_t numbersCount;
    cerr << "Enter numbers count: ";
    cin >> numbersCount;
    vector<double> numbers;
    bool generate;
    cerr << "Generate random numbers? (1/0): ";
    cin >> generate;
    if (generate) {
        srand(RAND_SEED);
        numbers = generateNumbersArray(numbersCount);
    } else {
        cerr << "Enter numbers: ";
        numbers = getNumbersArray(numbersCount);
    }
}
```



```

    }
    size_t binCount;
    cerr << "Enter bins count: ";
    cin >> binCount;

    vector<size_t> bins = getBinsDistribution(numbers,
binCount);
    size_t maxBinCount = findMaxBinCount(bins);
    cmdOutput(bins, maxBinCount);
}
vector<double> getNumbersArray(size_t numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        cin >> res[i];
    }
    return res;
}
vector<double> generateNumbersArray(size_t numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        //res[i] = rand() % 101;
        res[i] = getRandomNumber();
    }
    return res;
}
vector<size_t> getBinsDistribution(vector<double> numbers,
size_t binCount) {
    vector<size_t> res(binCount);
    double minVal = numbers[0];
    double maxVal = numbers[0];
    findMinMax(numbers, minVal, maxVal);
    double step = (maxVal - minVal) / binCount;

    for (auto item : numbers) {
        bool found = false;

```

```

        for (size_t i = 0; (i < binCount-1) && !found; i++)
        {
            auto lo = minVal + i * step;
            auto hi = minVal + (i + 1) * step;
            if ((lo <= item) && (item < hi)) {
                res[i]++;
                found = true;
            }
        }
        if (!found) {
            res[binCount - 1]++;
        }
    }
    return res;
}

void findMinMax(vector<double> numbers, double& min, double&
max) {
    for (int i = 1; i < numbers.size(); i++) {
        if (numbers[i] < min) {
            min = numbers[i];
        }
        else if (numbers[i] > max) {
            max = numbers[i];
        }
    }
}

void cmdOutput(vector<size_t> bins, size_t maxBinCount) {
    for (size_t item : bins) {
        cout << formatLabel(item) << "|" << formatGist(item,
maxBinCount) << endl;
    }
}

string formatLabel(size_t item) {
    string prefix = "";
    if (item / 10 == 0) {

```

```

        prefix = " ";
    } else if (item / 100 == 0) {
        prefix = " ";
    }
    return prefix + to_string(item);
}

string formatGist(size_t item, size_t maxBinCount) {
    if (maxBinCount <= MAX_GIST_WIDTH) {
        return string(item, GIST_CHAR);
    }
    else {
        size_t height = MAX_GIST_WIDTH *
(static_cast<double>(item) / maxBinCount);
        return string(height, GIST_CHAR);
    }
}

size_t findMaxBinCount(vector<size_t> bins) {
    size_t mx = 0;
    for (size_t item : bins) {
        if (item > mx) {
            mx = item;
        }
    }
    return mx;
}

double getRandomNumber() {
    double sm = 0;
    for (int i = 0; i < 12; i++) {
        sm += rand();
    }
    return sm;
}

```

## Спецификации функций

`vector<double> getNumbersArray(size_t numbersCount)`

– Функция ввода `numbersCount` чисел для заполнения массива. Возвращает массив из `numbersCount` элементов.

`NumbersCount` – Кол-во чисел в массиве

`double getRandomNumber()` – Генерация случайного числа.

Возвращает случайное число.

`vector<double> generateNumbersArray(size_t numbersCount)` – Заполнение массива значениями, которые генерирует функция `getRandomNumber()`. Возвращает массив из `numbersCount` элементов.

`vector<size_t> getBinsDistribution(vector<double> numbers, size_t binCount)` – Расчёт заполненности корзин.

`BinCount` – Кол-во корзин

Возвращает массив из `binCount` чисел – заполненности корзин.

`void findMinMax(vector<double> numbers, double& min, double& max)` – Поиск максимального и минимального числа в массиве. Не возвращает их, а передаёт по ссылке.

`size_t findMaxBinCount(vector<size_t> bins)` – Поиск максимально заполненной корзины.

`bins` – Массив заполненности корзин.

`void cmdOutput(vector<size_t> bins, size_t maxBinCount)` – Вывод в консоль

string formatLabel(size\_t item) – Форматирование подписей перед выводом.

item – Данные о заполненности одной корзины.

string formatGist(size\_t item, size\_t maxBinCount) – Форматирование гистограмм перед выводом.

Тестирование программы

Тестирование осуществляется с использованием следующих команд:

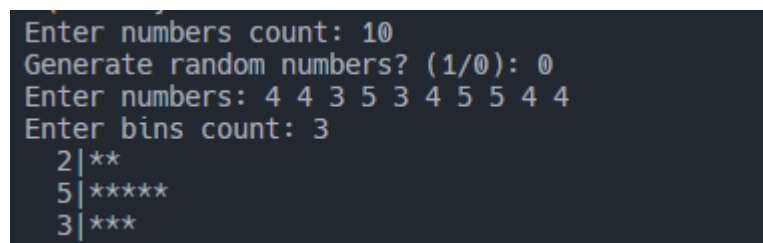
```
lab01.exe < 01-example.input.txt > 01-example.actual.txt
2>NUL
```

```
fc /N 01-example.actual.txt 01-example.expected.txt
```

```
lab01.exe < 02-example.input.txt > 02-example.actual.txt
2>NUL
```

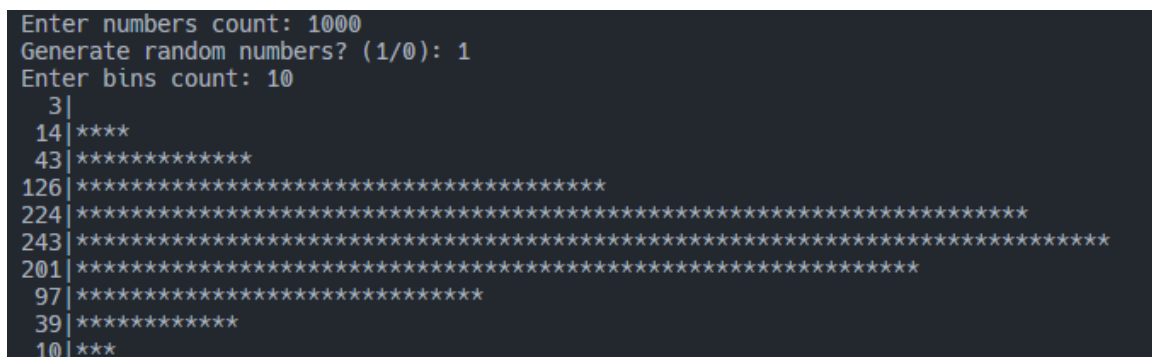
```
fc /N 02-example.actual.txt 02-example.expected.txt
```

Тестирование задачи представлено на рисунках 3-4.



```
Enter numbers count: 10
Generate random numbers? (1/0): 0
Enter numbers: 4 4 3 5 3 4 5 5 4 4
Enter bins count: 3
2|**
5|*****
3|***
```

Рисунок 3 – Тест 1 задачи лабораторной работы №1



```
Enter numbers count: 1000
Generate random numbers? (1/0): 1
Enter bins count: 10
3|
14|****
43|*****
126|*****
224|*****
243|*****
201|*****
97|*****
39|*****
10|***
```

Рисунок 4 – Тест 2 задачи лабораторной работы №1

## 2.2 Лабораторная работа №2. Система контроля версий Git

Команды, использовавшиеся во время выполнения задания, их описание и действия, выполненные над файлами проекта:

1. Создание папки `lab02` и вызов консоли Git Bush в ней из проводника.

```
mkdir alice  
mkdir bob  
ls  
cd alice
```

2. Создание папок `alice` и `bob`, отображение содержимого текущей папки (`lab02`) и переход в папку `alice`.

```
mkdir project  
cd project  
cd ..  
cd project
```

3. Создание папки `project`, переход в неё, назад на уровень выше в папку `alice` и опять в `project`.

```
git init  
git config user.name 'Alice (KozyavinMS)'  
git config user.email 'kzmaxim256@gmail.com'
```

4. Инициализация репозитория в текущей папке, настройка имени и электронной почты.

5. Создание файла `main.cpp` с заготовкой программы в папке `project`.

```
git status
```

6. Команда показывает, что в ветке `master` ещё не было произведено коммитов, есть неотслеживаемый файл `main.cpp` (рисунок 5).

```

$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        main.cpp

nothing added to commit but untracked files present (use "git add" to track)

```

Рисунок 5 – Реакция на команду `git status`

```

git add main.cpp
git status

```

7. После добавления файла в отслеживаемые команда показывает, что файл был изменён и его можно закоммитить (рисунок 6).

```

Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/alice/project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.cpp

```

Рисунок 6 – Реакция на команду `git status` после включения отслеживания файла

```

git commit -m 'code: заготовка программы'
git status

```

8. Создание коммита с сообщением 'code: заготовка программы'.

9. Добавление в `main.cpp` ввода двух чисел.

```

git add main.cpp
git commit -m "code: добавлен ввод двух чисел"

```

10. Добавление в `main.cpp` вывода суммы чисел.

```

git add -u
git commit -m "code: вывод суммы чисел"

```

11. Добавление в `main.cpp` вывода разности чисел.

```

git commit -a -m "code: вывод разности чисел"

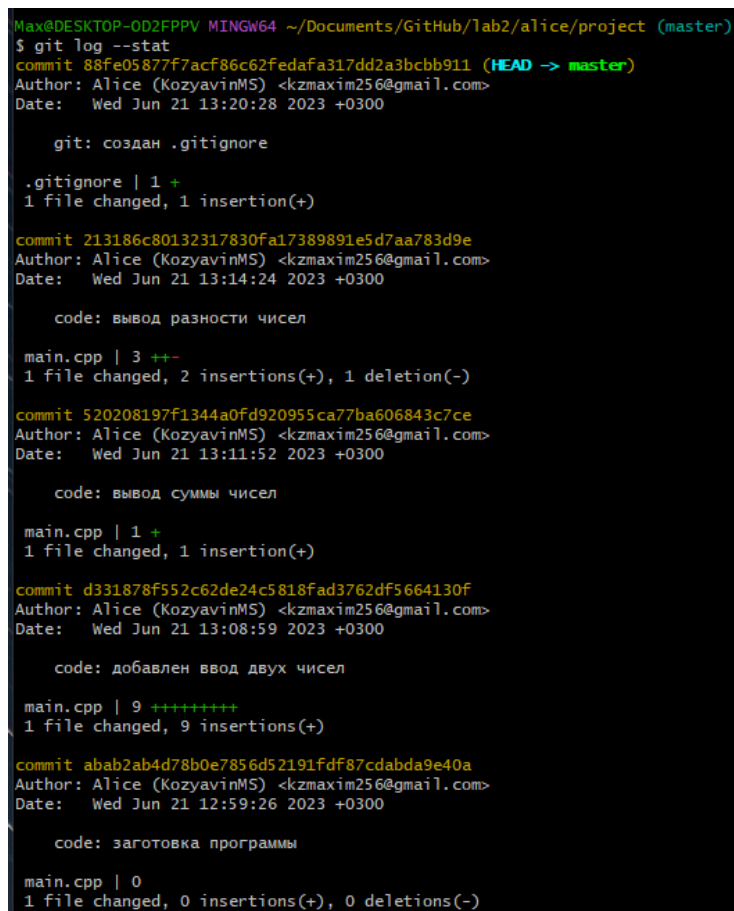
```

12. В шагах 8-11 произведены коммиты разными способами.

13. Создание файла `.gitignore` с указанием `main.exe` для того, чтобы игнорировать его при составлении коммитов.

```
git add .gitignore
git commit -m "git: создан .gitignore"
git log -stat
```

14. Команда `git log -stat` выводит историю коммитов, их описание, хэш, авторов и прочее (рисунок 7).



```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/alice/project (master)
$ git log --stat
commit 88fe05877f7acf86c62fedafa317dd2a3bcb911 (HEAD -> master)
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 13:20:28 2023 +0300

    git: создан .gitignore

.gitignore | 1 +
1 file changed, 1 insertion(+)

commit 213186c80132317830fa17389891e5d7aa783d9e
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 13:14:24 2023 +0300

    code: вывод разности чисел

main.cpp | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

commit 520208197f1344a0fd920955ca77ba606843c7ce
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 13:11:52 2023 +0300

    code: вывод суммы чисел

main.cpp | 1 +
1 file changed, 1 insertion(+)

commit d331878f552c62de24c5818fad3762df5664130f
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 13:08:59 2023 +0300

    code: добавлен ввод двух чисел

main.cpp | 9 ++++++++
1 file changed, 9 insertions(+)

commit abab2ab4d78b0e7856d52191fdf87cdabda9e40a
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 12:59:26 2023 +0300

    code: заготовка программы

main.cpp | 0
1 file changed, 0 insertions(+), 0 deletions(-)
```

Рисунок 7 – Реакция на команду `git log -stat`

```
git log --oneline --decorate
git log --oneline --decorate --all --graph
git log --grep "git:"
git log -- main.cpp
```

15. Опробована команда `git log` с разными параметрами.

```
git show HEAD~1
```



16. Просмотр информации предпоследнего коммита (рисунок 8).

```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/alice/project (master)
$ git show HEAD~1
commit 213186c80132317830fa17389891e5d7aa783d9e
Author: Alice (KozyavinMS) <kzmaxim256@gmail.com>
Date:   Wed Jun 21 13:14:24 2023 +0300

    code: вывод разности чисел

diff --git a/main.cpp b/main.cpp
index 8308a11..15b71a2 100644
--- a/main.cpp
+++ b/main.cpp
@@ -6,5 +6,6 @@ int main() {
     cout << "Enter A and B: ";
     int a, b;
     cin >> a >> b;
-    cout << a+b;
+    cout << "A + B = " << a + b << '\n'
+    << "A - B = " << a - b << '\n';
 }
\ No newline at end of file
```

Рисунок 8 – Реакция на команду `git show HEAD~1`

```
git show master~1
```

```
git show 213186c
```

17. Просмотр предпоследнего коммита несколькими способами.

18. Добавление в `main.cpp` вывода произведения чисел.

```
git diff
```

19. Вывод отличий текущего и предыдущего коммита (рисунок 9).

```
$ git diff
diff --git a/main.cpp b/main.cpp
index 15b71a2..8508db0 100644
--- a/main.cpp
+++ b/main.cpp
@@ -7,5 +7,6 @@ int main() {
     int a, b;
     cin >> a >> b;
     cout << "A + B = " << a + b << '\n'
-    << "A - B = " << a - b << '\n';
+    << "A - B = " << a - b << '\n'
+    << "A * B = " << a * b << '\n';
 }
\ No newline at end of file
```

Рисунок 9 – Реакция на команду `git diff`

```
git diff HEAD~2
```

```
git diff HEAD~2 HEAD
```

```
git diff abab2ab4 213186c
```

20. Вывод разными способами различий между указанными коммитами.

```
git commit -a -m "code: вывод произведения"  
git reset --hard HEAD~1
```

21. Отправка коммита и его полный откат.

22. Добавление комментария в main.cpp.

```
git checkout HEAD -- main.cpp
```

23. Восстановление файла до состояния последнего коммита.

```
ssh-keygen
```

24. Создание закрытого ключа, добавление его на GitHub как ключ для внесения изменений в репозиторий.

```
git remote add origin git@github.com:Kanzu32/lab2.git  
git branch -M main  
git push -u origin main
```

25. Загрузка проекта на удалённое хранилище.

Переходим на другую машину. (Создаём новый терминал в папке bob)

```
git clone git@github.com:Kanzu32/lab2.git project  
cd project  
git config user.name "Bob (KozyavinMS)"  
git config user.email 'kzmaxim256@gmail.com'
```

26. Копирование проекта из удалённого хранилища и настройка данных пользователя.

27. Добавление в main.cpp вывода произведения.

```
git commit -a -m "боб коммитит умножение code:)"  
git push
```

28. Коммит изменений.

Переходим на машину Алисы. (Первый терминал)

```
git fetch  
git log --oneline --decorate --all --graph  
git pull --ff-only
```

29. Получение изменений, просмотр и обновление локальных файлов до последнего коммита.

30. Добавление в main.cpp вывода частного.

```
git commit -a -m "code: деление"
git push
```

Переходим на машину Боба. (Второй терминал)

```
git fetch
git log --oneline --decorate --all --graph
git pull --ff-only
```

Переходим на машину Алисы. (Первый терминал)

31. Добавление в main.cpp вывода минимального числа.

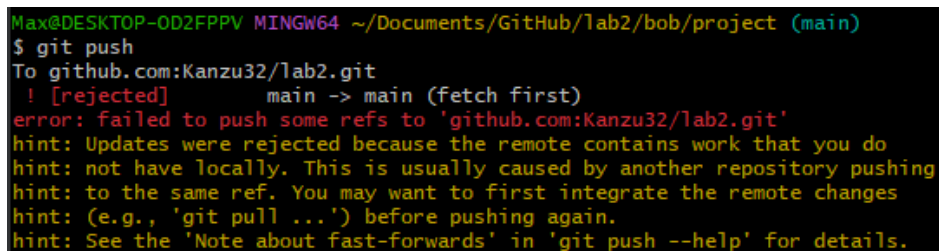
```
git commit -a -m "code: минимум"
git push
```

Переходим на машину Боба. (Второй терминал)

32. Добавление в main.cpp вывода максимального числа.

```
git commit -a -m "code: максимум"
git push
```

33. Отправить коммит на удалённый сервер не выходит т.к. не был получен последний коммит с удалённого хранилища (рисунок 10).



```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/bob/project (main)
$ git push
To github.com:Kanzu32/lab2.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'github.com:Kanzu32/lab2.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Рисунок 10 – Отклонение отправки коммита на удалённое хранилище

```
git fetch
git log --oneline --decorate --all --graph
git rebase origin/main
```

34. Получаем изменения, совмещаем локальный коммит и коммит из удалённого хранилища. Получаем конфликт (рисунок 11). Необходимо вручную исправить конфликт (рисунок 12).

```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/bob/project (main)
$ git rebase origin/main
Auto-merging main.cpp
CONFLICT (content): Merge conflict in main.cpp
error: could not apply b43f1f0... code: максимум
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply b43f1f0... code: максимум
```

Рисунок 11 – Конфликт слияния

```
12 | << "A / B = " << a / b << '\n'
    | Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
13 | <<<<<<< HEAD (Current Change)
    | << "MIN: " << min(min(min(a + b, a - b), a * b), a / b)
14 |
    | =====
15 | << "MAX: " << max(max(max(a + b, a - b), a * b), a / b)
16 |
    | <>>>>>> b43f1f0 (code: максимум) (Incoming Change)
17 |
    | }
18 |
```

Рисунок 12 – Автоматическое обозначение конфликтующих строк

```
git add main.cpp
git rebase --continue
git log --oneline --decorate --all --graph
git push
```

35. После изменений загрузка на удалённое хранилище выполняется.

Переходим на машину Алисы. (Первый терминал)

```
git branch double
git checkout double
```

36. Создаём новую ветку double и переключаемся на неё.

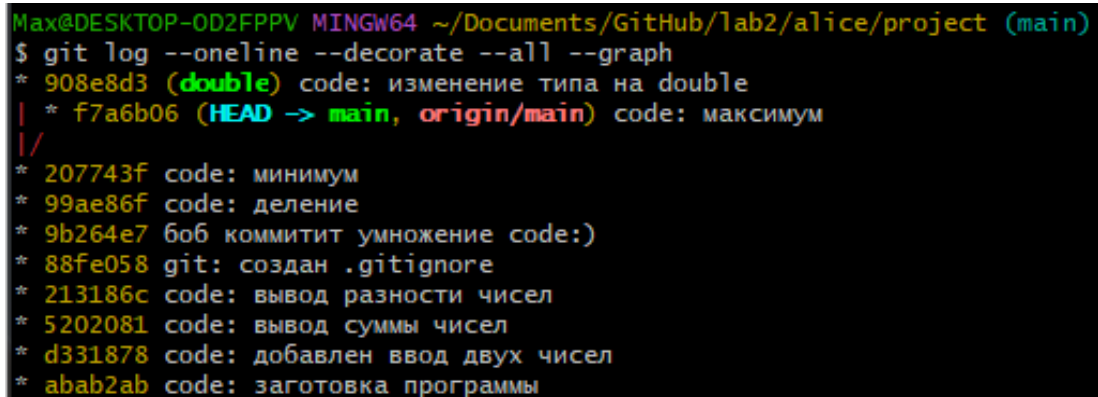
37. Изменение типа данных в файле main.cpp.

```
git commit -m "code: изменение типа на double"
git checkout main
```

```
git fetch
```

```
git log --oneline --decorate --all --graph
```

38. Коммитим в отдельную ветку и переключаемся на main. Видим разветвление в списке коммитов (рисунок 13).



```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/alice/project (main)
$ git log --oneline --decorate --all --graph
* 908e8d3 (double) code: изменение типа на double
| * f7a6b06 (HEAD -> main, origin/main) code: максимум
|/
* 207743f code: минимум
* 99ae86f code: деление
* 9b264e7 боб коммитит умножение code:)
* 88fe058 git: создан .gitignore
* 213186c code: вывод разности чисел
* 5202081 code: вывод суммы чисел
* d331878 code: добавлен ввод двух чисел
* abab2ab code: заготовка программы
```

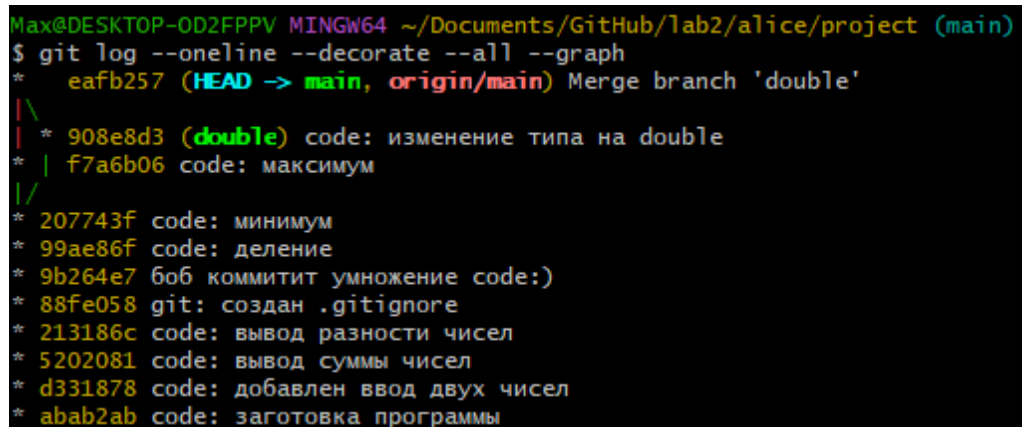
Рисунок 13 – Две ветки коммитов

```
git merge double
```

```
git push
```

```
git log --oneline --decorate --all --graph
```

39. Сливаем ветки и коммитим. Получаем программу с типом переменных double и функцией максимума (рисунок 14).



```
Max@DESKTOP-OD2FPPV MINGW64 ~/Documents/GitHub/lab2/alice/project (main)
$ git log --oneline --decorate --all --graph
* eafb257 (HEAD -> main, origin/main) Merge branch 'double'
| \
| * 908e8d3 (double) code: изменение типа на double
| * f7a6b06 code: максимум
|/
* 207743f code: минимум
* 99ae86f code: деление
* 9b264e7 боб коммитит умножение code:)
* 88fe058 git: создан .gitignore
* 213186c code: вывод разности чисел
* 5202081 code: вывод суммы чисел
* d331878 code: добавлен ввод двух чисел
* abab2ab code: заготовка программы
```

Рисунок 14 – Финальный вид истории коммитов

## 2.3 Лабораторная работа №3. Декомпозиция программы функциями

### Текст программы

```
//main.cpp

#include <iostream>
#include <vector>
#include <ctime>
#include "minmax.h"
#include "minmax.cpp"
#include "svg.h"
#include "svg.cpp"
using namespace std;
const char GIST_CHAR = '*';
const size_t SCREEN_WIDTH = 80;
const size_t MAX_GIST_WIDTH = SCREEN_WIDTH - 3 - 1;
const auto RAND_SEED = time(0);
const bool SVG_OUTPUT = true;

vector<double> getNumbersArray(size_t numbersCount);
vector<double> generateNumbersArray(size_t numbersCount);
vector<size_t> getBinsDistribution(vector<double> numbers,
size_t binCount);
void cmdOutput(vector<size_t> bins, size_t maxBinCount);
string formatLabel(size_t item);
string formatGist(size_t item, size_t maxBinCount);
size_t findMaxBinCount(vector<size_t> bins);
double getRandomNumber();

int main() {
    srand(RAND_SEED);
    size_t numbersCount;
    cerr << "Enter numbers count: ";
    cin >> numbersCount;
    vector<double> numbers;
    bool generate;
    cerr << "Generate random numbers? (1/0): ";
```

```

    cin >> generate;
    if (generate) {
        numbers = generateNumbersArray(numbersCount);
    } else {
        cerr << "Enter numbers: ";
        numbers = getNumbersArray(numbersCount);
    }
    size_t binCount;
    cerr << "Enter bins count: ";
    cin >> binCount;
    vector<size_t> bins = getBinsDistribution(numbers,
binCount);
    size_t maxBinCount = findMaxBinCount(bins);

    if (SVG_OUTPUT) {
        svgOutput(bins, maxBinCount);
    } else {
        cmdOutput(bins, maxBinCount);
    }
}

vector<double> getNumbersArray(size_t numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        cin >> res[i];
    }
    return res;
}

vector<double> generateNumbersArray(size_t numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        //res[i] = rand() % 101;
        res[i] = getRandomNumber();
    }
    return res;
}

```

```

        vector<size_t>  getBinsDistribution(vector<double>  numbers,
size_t binCount) {
    vector<size_t> res(binCount);
    double minVal = 0;
    double maxVal = 0;
    findMinMax(numbers, minVal, maxVal);
    double step = (maxVal - minVal) / binCount;
    for (auto item : numbers) {
        bool found = false;
        for (size_t i = 0; (i < binCount-1) && !found; i++)
        {
            auto lo = minVal + i * step;
            auto hi = minVal + (i + 1) * step;
            if ((lo <= item) && (item < hi)) {
                res[i]++;
                found = true;
            }
        }
        if (!found) {
            res[binCount - 1]++;
        }
    }
    return res;
}

string formatLabel(size_t item) {
    string prefix = "";
    if (item / 10 == 0) {
        prefix = " ";
    } else if (item / 100 == 0) {
        prefix = " ";
    }
    return prefix + to_string(item);
}

string formatGist(size_t item, size_t maxBinCount) {
    if (maxBinCount <= MAX_GIST_WIDTH) {

```



```

        return string(item, GIST_CHAR);
    }
    else {
        size_t      height      =      MAX_GIST_WIDTH      *
(static_cast<double>(item) / maxBinCount);
        return string(height, GIST_CHAR);
    }
}

size_t findMaxBinCount(vector<size_t> bins) {
    size_t mx = 0;
    for (size_t item : bins) {
        if (item > mx) {
            mx = item;
        }
    }
    return mx;
}

double getRandomNumber() {
    double sm = 0;
    for (int i = 0; i < 12; i++) {
        sm += rand();
    }
    return sm;
}

void cmdOutput(vector<size_t> bins, size_t maxBinCount) {
    for (size_t item : bins) {
        cout << formatLabel(item) << "|" << formatGist(item,
maxBinCount) << endl;
    }
}

```

**//minmax.h**

```

#ifndef MINMAX_H
#define MINMAX_H

```

```

#include <vector>
using namespace std;
void findMinMax(const vector<double>& numbers, double& min,
double& max);
#endif

```

**//minmax.cpp**

```

#include "minmax.h"
void findMinMax(const vector<double>& numbers, double& min,
double& max) {
    if (numbers.size() > 0) {
        min = numbers[0];
        max = numbers[0];
    } else {
        return;
    }
    for (int i = 1; i < numbers.size(); i++) {
        if (numbers[i] < min) {
            min = numbers[i];
        }
        else if (numbers[i] > max) {
            max = numbers[i];
        }
    }
}

```

**//svg.h**

```

#ifndef SVG_H
#define SVG_H
#include <vector>
#include <string>
#include <iostream>

using namespace std;

```

```

    const vector<string> COLORS {"#7FFF00", "#DC143C", "#1E90FF",
"#FFA500", "#8B008B"};
    int lastColorIndex = -1;
    const bool OPACITY_CHANGE = true;
    const bool USE_STROKE = false;
    const size_t SVG_MAX_GIST_WIDTH = 80;

    const string STROKE_COLOR = "black";
    const auto STROKE_WIDTH = 2;
    const auto TEXT_X = 20;
    const auto TEXT_Y = 20;
    const auto TEXT_WIDTH = 60;
    const auto BIN_HEIGHT = 30;
    const auto BLOCK_WIDTH = 15;

    void svgOutput(vector<size_t> bins, size_t maxBinCount);
    void svgHeader(double width, double height);
    void svgFooter();
    void svgText(double x, double y, string text);
    void svgRect(double x, double y, double width, double height,
string color, double opacity);
#endif

```

**//svg.cpp**

```

#include "svg.h"
string getRandomColor() {
    size_t index = rand() % COLORS.size();
    if (index == lastColorIndex) {
        index = (lastColorIndex + 1) % COLORS.size();
    }
    lastColorIndex = index;
    return COLORS[index];
}
double getOpacity(size_t bin, size_t maxBinCount) {
    if (maxBinCount == 0) {

```

```

        return 0;
    }
    double opacity = bin / static_cast<double>(maxBinCount);
    if (opacity < 0.2) opacity = 0.2;
    return opacity;
}

void svgOutput(vector<size_t> bins, size_t maxBinCount) {
    double coeff = 1;
    if (maxBinCount > SVG_MAX_GIST_WIDTH) {
        coeff = SVG_MAX_GIST_WIDTH /
static_cast<double>(maxBinCount);
    }
    svgHeader((TEXT_X + TEXT_WIDTH) + maxBinCount * coeff *
BLOCK_WIDTH + STROKE_WIDTH*USE_STROKE, bins.size()*BIN_HEIGHT);
    double top = 0;
    for (size_t bin : bins) {
        const double bin_width = BLOCK_WIDTH * bin * coeff;
        double opacity = 1;
        if (OPACITY_CHANGE) opacity = getOpacity(bin,
maxBinCount);
        svgText(TEXT_X, top + TEXT_Y, to_string(bin));
        svgRect(TEXT_WIDTH, top, bin_width, BIN_HEIGHT,
getRandomColor(), opacity);
        top += BIN_HEIGHT;
    }
    svgFooter();
}

void svgHeader(double width, double height) {
    cout << "<?xml version='1.0' encoding='UTF-8' ?>\n"
<< "<svg " << "width='" << width << "' height='" << height
<< "' viewBox='0 0 " << width << " " << height
<< " ' xmlns='http://www.w3.org/2000/svg'>\n";
}

void svgText(double x, double y, string text) {

```

```

        cout << "<text x='" << x << "' y='" << y << "' style='font:
bold 18px sans-serif;'" << text << "</text>";
    }

    void svgRect(double x, double y, double width, double height,
string color, double opacity) {
        cout << "<rect x='" << x << "' y='" << y << "' width='"
<< width << "' height='" << height << "' fill='" << color;
        if (USE_STROKE) {
            cout << "' stroke='" << STROKE_COLOR << "' stroke-
width='" << STROKE_WIDTH;
        }
        cout << "' fill-opacity='" << opacity << "'/>";
    }

    void svgFooter() {
        cout << "</svg>\n";
    }

```

### Спецификации функций

Спецификации функций, описанных в лабораторной 1, остаётся такой же.

`string getRandomColor()` – возвращает случайный цвет из константного массива `COLORS`.

`double getOpacity(size_t bin, size_t maxBinCount)` – принимает заполненность корзины и максимальную заполненность из всех корзин, возвращает прозрачность корзины.

`void svgOutput(vector<size_t> bins, size_t maxBinCount)` – функция выводящая данные в формате SVG. Вызывает соответствующие функции для генерации XML разметки.

`void svgHeader(double width, double height)` – функция описывающая заголовок SVG файла с указанием высоты и ширины.

`void svgText(double x, double y, string text)` – функция описывающая текст в SVG файле по координатам  $x$  и  $y$ .

`void svgRect(double x, double y, double width, double height, string color, double opacity)` – функция описывающая прямоугольник в SVG файле по координатам  $x$  и  $y$ , ширине, высоте, цвету и прозрачности.

`void svgFooter()` – функция закрывающая тег тела SVG файла, открытый в функции `svgHeader()`.

### Тестирование программы

Входные данные:

1000

1

10

Сид генерации случайных чисел: 42

Выходные данные:

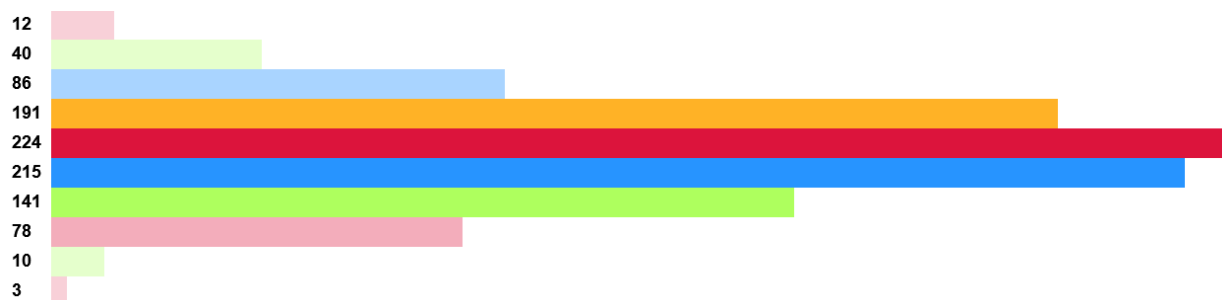


Рисунок 15 – Вывод данных в формате SVG со случайным заполнением

Входные данные:

10

0

4 4 3 5 3 4 5 5 4 4

3

Выходные данные представлены на рисунке 16.



Рисунок 16 – Вывод данных в формате SVG введенных пользователем

### Unit-тесты

```
#include <cassert>
```

```
#include "minmax.h"
```

```
#include "minmax.cpp"
```

```
#include "svg.h"
```

```
#include "svg.cpp"
```

```
void testMinMaxPositive();
```

```
void testMinMaxNegative();
```

```
void testMinMaxOne();
```

```
void testMinMaxZero();
```

```
void testOpacity();
```

```
void testOpacityLow();
```

```
int main() {
```

```
    testMinMaxPositive();
```

```
    testMinMaxNegative();
```

```
    testMinMaxOne();
```

```
    testMinMaxZero();
```

```
    testOpacity();
```

```
    cout << "UNIT-TESTS: OK" << endl;
```

```

}

void testOpacity() {
    double res = getOpacity(5, 10);
    assert(res == 0.5);
}

void testOpacityZero() {
    double res = getOpacity(5, 0);
    assert(res == 0);
}

void testOpacityLow() {
    double res = getOpacity(5, 1000);
    assert(res == 0.2);
}

void testMinMaxPositive() {
    double min = 0;
    double max = 0;
    findMinMax(vector<double>{1, 2, 3}, min, max);
    assert(min == 1);
    assert(max == 3);
}

void testMinMaxNegative() {
    double min = 0;
    double max = 0;
    findMinMax(vector<double>{-1, -2, -3}, min, max);
    assert(min == -3);
    assert(max == -1);
}

void testMinMaxOne() {
    double min = 0;
    double max = 0;
    findMinMax(vector<double>{3}, min, max);
    assert(min == 3);
    assert(max == 3);
}

void testMinMaxZero() {

```



```

    double min = 0;
    double max = 0;
    findMinMax(vector<double>{}, min, max);
    assert(min == 0);
    assert(max == 0);
}

```

### Описание тестов

Unit-тесты функции `double getOpacity(size_t bin, size_t maxBinCount)` принимающей заполненность переданной корзины и максимальную заполненность корзины и возвращающей прозрачность переданной корзины для отображения в SVG формате. Если возвращаемое значение меньше 0.2, то вместо него возвращают 0.2.

`testOpacity()` – Тест при нормальных значениях `bin` и `maxBinCount`.

`testOpacityZero()` – Тест при значении `maxBinCount = 0`.

`testOpacityLow()` – Тест при возвращаемом значении меньше 0.2.

Unit-тесты функции `void findMinMax(const vector<double>& numbers, double& min, double& max)` принимающей массив чисел и минимальное и максимальное значение по ссылкам для их записи.

`testMinMaxPositive()` – Тест при положительных значениях чисел в массиве.

`testMinMaxNegative()` – Тест при отрицательных значениях чисел в массиве.

`testMinMaxOne()` – Тест при передаче массива с одним элементом.

`testMinMaxZero()` – Тест при передаче пустого массива.

## 2.4 Лабораторная работа №4. Библиотеки

### Текст программы

```
//main.cpp

#include <iostream>
#include <vector>
#include <ctime>
#include <sstream>
#include "modules/minmax.h"
#include "modules/minmax.cpp"
#include "modules/svg.h"
#include "modules/svg.cpp"
#include "curl/include/curl/curl.h"
using namespace std;
const char GIST_CHAR = '*';
const size_t SCREEN_WIDTH = 80;
const size_t MAX_GIST_WIDTH = SCREEN_WIDTH - 3 - 1;
const auto RAND_SEED = time(0);
const bool SVG_OUTPUT = true;
struct Input {
    vector<double> numbers;
    size_t binCount;
};
Input getInput(istream& in, bool prompt);
vector<double>      getNumbersArray(istream&      in,      size_t
numbersCount);
vector<double> generateNumbersArray(size_t numbersCount);
vector<size_t> getBinsDistribution(Input& data);
void cmdOutput(vector<size_t> bins, size_t maxBinCount);
string formatLabel(size_t item);
string formatGist(size_t item, size_t maxBinCount);
size_t findMaxBinCount(vector<size_t> bins);
double getRandomNumber();
size_t writeInBuffer(void* items, size_t itemSize, size_t
itemCount, void* ctx);
Input download(const string& address);
```

```

int main(int argc, char* argv[]) {
    srand(RAND_SEED);
    Input data;
    if (argc > 1) {
        data = download(argv[1]);
    } else {
        data = getInput(cin, true);
    }
    cerr << data.binCount;
    vector<size_t> bins = getBinsDistribution(data);
    size_t maxBinCount = findMaxBinCount(bins);
    if (SVG_OUTPUT) {
        svgOutput(bins, maxBinCount);
    } else {
        cmdOutput(bins, maxBinCount);
    }
}

Input download(const string& address) {
    stringstream buffer;
    CURL* curl = curl_easy_init();
    if (curl) {
        CURLcode res;
        curl_easy_setopt(curl, CURLOPT_URL,
address.c_str());
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION,
writeInBuffer);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &buffer);
        curl_easy_setopt(curl, CURLOPT_FAILONERROR, 1);
        res = curl_easy_perform(curl);
        if (res == CURLE_HTTP_RETURNED_ERROR) {
            cerr << "FAILED TO LOAD FILE" << endl;
        }
        if (res != CURLE_OK) {
            cerr << "ERROR " << res << ": " <<
curl_easy_strerror(res) << endl;

```

```

        exit(1);
    }
    curl_easy_cleanup(curl);
}
return getInput(buffer, false);
}

size_t writeInBuffer(void* items, size_t itemSize, size_t
itemCount, void* ctx) {
    stringstream* buffer =
reinterpret_cast<stringstream*>(ctx);
    const char* data = reinterpret_cast<char*>(items);
    size_t dataSize = itemCount * itemSize;
    buffer->write(data, dataSize);
    return dataSize;
}

Input getInput(istream& in, bool prompt) {
    Input data;
    size_t numbersCount;
    if (prompt) {cerr << "Enter numbers count: ";};
    in >> numbersCount;

    bool generate;
    if (prompt) {cerr << "Generate random numbers? (1/0):
";};

    in >> generate;
    if (generate) {
        data.numbers = generateNumbersArray(numbersCount);
    } else {
        cerr << "Enter numbers: ";
        data.numbers = getNumbersArray(in, numbersCount);
    }
    if (prompt) {cerr << "Enter bins count: ";};
    in >> data.binCount;
    return data;
}

```

```

        vector<double>      getNumbersArray(istream&      in,      size_t
numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        in >> res[i];
    }
    return res;
}

vector<double> generateNumbersArray(size_t numbersCount) {
    vector<double> res(numbersCount);
    for (int i = 0; i < numbersCount; i++) {
        //res[i] = rand() % 101;
        res[i] = getRandomNumber();
    }
    return res;
}

vector<size_t> getBinsDistribution(Input& data) {
    vector<size_t> res(data.binCount);
    double minVal = 0;
    double maxVal = 0;
    findMinMax(data.numbers, minVal, maxVal);
    double step = (maxVal - minVal) / data.binCount;
    for (auto item : data.numbers) {
        bool found = false;
        for (size_t i = 0; (i < data.binCount-1) && !found;
i++) {

            auto lo = minVal + i * step;
            auto hi = minVal + (i + 1) * step;
            if ((lo <= item) && (item < hi)) {
                res[i]++;
                found = true;
            }
        }
        if (!found) {
            res[data.binCount - 1]++;

```

```

        }
    }
    return res;
}

string formatLabel(size_t item) {
    string prefix = "";
    if (item / 10 == 0) {
        prefix = " ";
    } else if (item / 100 == 0) {
        prefix = " ";
    }
    return prefix + to_string(item);
}

string formatGist(size_t item, size_t maxBinCount) {
    if (maxBinCount <= MAX_GIST_WIDTH) {
        return string(item, GIST_CHAR);
    }
    else {
        size_t height = MAX_GIST_WIDTH *
(static_cast<double>(item) / maxBinCount);
        return string(height, GIST_CHAR);
    }
}

size_t findMaxBinCount(vector<size_t> bins) {
    size_t mx = 0;
    for (size_t item : bins) {
        if (item > mx) {
            mx = item;
        }
    }
    return mx;
}

double getRandomNumber() {
    double sm = 0;
    for (int i = 0; i < 12; i++) {

```

```

        sm += rand();
    }
    return sm;
}

void cmdOutput(vector<size_t> bins, size_t maxBinCount) {
    for (size_t item : bins) {
        cout << formatLabel(item) << "|" << formatGist(item,
maxBinCount) << endl;
    }
}

```

Модули `svg.h`, `svg.cpp`, `minmax.h`, `minmax.cpp` не изменены при рефакторинге и не затрагивают часть программы, отвечающую за ввод данных по url ссылке, поэтому остались такими же, как и в лабораторной 3.

### Рефакторинг

Массив чисел `vector<double> numbers` и кол-во корзин `size_t binCount` объединены в одну структуру данных `Input`.

```

struct Input {
    vector<double> numbers;
    size_t binCount;
};

```

Весь процесс получения данных от пользователя и случайной генерации данных объединён в функцию `Input getInput(istream& in, bool prompt)`.

Функция `vector<size_t> getBinsDistribution(Input& data)` теперь принимает только один параметр – новую структуру данных `Input`.

Функция `vector<double> getNumbersArray(istream& in, size_t numbersCount)` теперь принимает поток ввода поскольку может происходить считывание из разных потоков.

### Спецификации функций:

Часть функций остаются такими же, как и в лабораторной 3.

`Input getInput(istream& in, bool prompt)` – функция ввода данных

Входные данные:

`in` – поток ввода данных. Стандартный ввод или текстовый поток из скачанного файла.

`prompt` – разрешение или запрет вывода подсказок.

Выходные данные:

Структура данных `Input` содержащая массив чисел и количество корзин.

`Input download(const string& address)` – функция скачивающая файл по адресу с помощью библиотеки `Curl`. Возвращает структуру данных `Input`.

`size_t writeInBuffer(void* items, size_t itemSize, size_t itemCount, void* ctx)` – функция записывающая считанные данные в буфер.

Входные данные:

`items` – указатель на принятые данные.

`itemSize` – размер одного блока данных.

`itemCount` – количество блоков данных.

`ctx` – пользовательские данные (контекст). В данном случае буфер для считывания данных.

Выходные данные:

`dataSize` – размер принятых данных в байтах.



## Сборка программы

Для корректной работы программы необходим файл `libcurl-x64.dll` или `libcurl-x32.dll`. Его можно добавить в папку с проектом или добавить в PATH в настройках среды операционной системы. Так же локально должны быть файлы библиотеки Curl.

Для сборки программы необходимо прописать в терминале следующую команду:

```
g++ -Lcurl/lib lab04.cpp -lcurl -o lab04.exe
```

## Тестирование программы

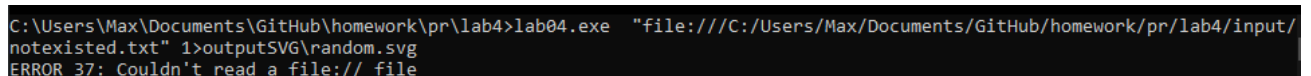
Тест 1. Попытка передать несуществующий файл.

Запуск из терминала:

```
lab04.exe 1> outputSVG\random.svg  
"file:///C:/Users/Max/Documents/GitHub/homework/pr/lab4/input/notexisted.txt"
```

Входные данные: Отсутствуют

На рисунке 17 представлена реакция программы на передачу несуществующего файла.



```
C:\Users\Max\Documents\GitHub\homework\pr\lab4>lab04.exe "file:///C:/Users/Max/Documents/GitHub/homework/pr/lab4/input/notexisted.txt" 1>outputSVG\random.svg  
ERROR 37: Couldn't read a file:// file
```

Рисунок 17 – Ошибка при передаче несуществующего файла

Тест 2. Попытка передать страницу выдающую ошибку. (любой код выше либо равный 400)

Запуск из терминала:

```
lab04.exe 1> outputSVG\random.svg  
"http://google.com/gfgfdgdf"
```

Тест соответствует 17 варианту индивидуального задания, заключающимся в отказе обрабатывать сайт, выдающий ошибку.

Входные данные: Адрес сайта, выдающего ошибку 404.

На рисунке 18 представлена реакция программы при выдаче ошибки сайтом.

```
C:\Users\Max\Documents\GitHub\homework\pr\lab4>lab04.exe "http://google.com/gfgfdgdf" 1>outputSVG\random.svg  
FAILED TO LOAD FILE  
ERROR 22: HTTP response code said error
```

Рисунок 18 – Отмена чтения файла при выдаче ошибки сайтом

### Тест 3. Передача нормального файла.

Запуск из терминала:

```
lab04.exe 1> outputSVG\curl.svg 2>NUL  
"file:///C:/Users/Max/Documents/GitHub/homework/pr/lab4/input/example.txt"
```

Входные данные: файл example.txt содержащий следующие данные:

```
10  
0  
4 4 3 5 3 4 5 5 4 4  
3
```

На рисунке 19 представлен вывод файла в SVG.



Рисунок 19 – Вывод в SVG файл при корректной URL ссылке

### 3 Рабочий график проведения практики

План проведения практики представлен в таблице 1.

Таблица 1 – План проведения практики

№ п/п	Содержание (типовые задания для текущего контроля)	Дата
1	Установочная конференция по практике. Составление индивидуального плана работы.	19.06.2023
2	Выполнение лабораторной работы 1. Вывод гистограммы в консоль.	20.06.2023
3	Выполнение лабораторной работы 1. Масштабирование гистограммы в зависимости от размера.	21.06.2023
4	Подготовка отчёта по лабораторной работе 1.	22.06.2023
5	Настройка и изучение системы контроля версий Git.	23.06.2023
6	Выполнение лабораторной работы 2.	24.06.2023
7	Подготовка отчёта по лабораторной работе 2.	26.06.2023
8	Защита лабораторной 1.	27.06.2023
9	Выполнение лабораторной работы 3. Подготовка отчёта по лабораторной работе 3.	28.06.2023
10	Защита лабораторной 2.	29.06.2023
11	Защита лабораторной 3.	30.06.2023
12	Выполнение лабораторной работы 4.	01.07.2023
13	Подготовка отчёта по лабораторной работе 4.	03.07.2023
14	Защита лабораторной 4.	04.07.2023
15	Подготовка отчёта по практике.	05.07.2023
16	Итоговая конференция по практике.	06.07.2023

## ЗАКЛЮЧЕНИЕ

В ходе эксплуатационной практики были получены знания, а также приобретены умения и навыки, требуемые для работы с системой контроля версий Git и языком программирования C++. В ходе выполнения лабораторных работ была осуществлена практическая отработка умений программирования, структуризации и рефакторинга программы и совместной работы над проектом. Также отрабатывались умения по работе с файлами SVG и их форматирования, работе с сетью и со сторонними библиотеками.

При выполнении практических заданий изучалась и применялась командная строка Git Bash и работа с BAT файлами.

На базе языка программирования C++ разработаны приложения для отрисовки гистограмм в терминал и в SVG файл с использованием различных способов ввода данных.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация языка C++ структура данных vector [Сайт]. URL: <https://en.cppreference.com/w/cpp/container/vector> (дата обращения: 21.06.2023).
2. Документация по работе с Git [Сайт]. URL: <https://git-scm.com/docs> (дата обращения: 23.06.2023).
3. Описание формата SVG [Сайт]. URL: <https://developer.mozilla.org/en-US/docs/Web/SVG> (дата обращения: 25.06.2023).
4. Документация библиотеки Curl. Описание параметра `CURLOPT_FAILONERROR` [Сайт]. URL: [https://curl.se/libcurl/c/CURLOPT\\_FAILONERROR.html](https://curl.se/libcurl/c/CURLOPT_FAILONERROR.html) (дата обращения: 28.06.2023).